Disclaimer: Este apunte no es autocontenido y fue pensado como un repaso de los conceptos, no para aprenderlos de aquí directamente.

Índice

1. Redes de computadoras

Son de propósito general, no específicas para un tipo de dato (contrario a TV, teléfonos, etc).

Los requerimientos dependen del punto de vista. Programador: Servicios de red copados. Diseñador: Bajo costo y eficiencia del hard. Admin: Facilicidad de mantenimiento.

Conectividad: Diseño escalable (no siempre es necesario, hay redes intrínsecamente chicas)

- Conmutación de circuitos: Se arma un camino para cada conexión.
- Conmutación de paquetes: Manda paquetes de un host a otro y no importa el camino.
  Store and forward: Espera todo el paquete y luego lo manda

Switch: Solo forwardea paquetes, no mantiene estados. Router: Conectan distintas redes. Cada red tiene direcciones para cada host, broadcast y posiblemente multicast.

Links multiplexados: Pueden ser usados por varias comunicaciones al mismo tiempo (se comparte).

- STDM: Sync Time Div Multiplex, divide tiempo en quantums y hace round robbin
  Problema: Se desperdicia tiempo.
- FDM: Freq Div Multiplex: Transmite todos juntos en distintas frecuencias (bandas).
  Problema: Hay que limitar la cantidad de flujos en el link.
- Statistical Multiplexing: STDM pero cada flujo manda en cualq quantum (tipo scheduling). Tamaño max para evitar monopolio (en gral de 1 paquete).

Congestión: Un nodo recibe mas de lo que puede procesar => Buffer lleno => Drop de paquetes

Canal lógico: Request/reply o msg stream. Inteligencia de canales puede ir a hosts o a switches.

Reliability: Servicio importante. Errores:

- Bits corruptos (en general una tira consecutiva)
- Pérdida de paquetes (distinguir no conectado de lento)
- Nodos o links que se caen

Semantic Gap: En medio de la app y la realidad física de la red: Reducirlo por soft.

2. Modelos OSI y TCP/IP

Arquitectura: Forma del grafo de protocolos.
Protocolo: Implementación de una capa. Interfase hacia arriba (capa superior) y hacia su contraparte en la otra máquina.

3. Teoría de la información

3.1. Entropía

Si p_i es la probabilidad de ocurrencia del suceso s_i, se define la *información* del suceso s_i como:

$$I_i = log_2\left(\frac{1}{p_i}\right)$$

Si tengo una fuente S de información, con sucesos $s_1 \dots s_n$ de probabilidad $p_1 \dots p_n$, se define la *entropía* de la fuente como:

$$H(S) = \sum_{i=1}^{n} p_i \times I_i$$

Es decir, es la esperanza de la información. Invirtiendo el logaritmo, bajando el exponente y sacando factor común el -1 podemos obtener una expresión equivalente, expresada de manera más cómoda:

$$H(S) = \sum_{i=1}^{n} p_i \times I_i = \sum_{i=1}^{n} p_i \times log_2\left(\frac{1}{p_i}\right) = \sum_{i=1}^{n} p_i \times -log_2\left(p_i\right) = -\sum_{i=1}^{n} p_i \times log_2\left(p_i\right)$$

3.2. Códigos

La extensión de orden n de una fuente S, S^n, es considerar como suceso cualquier ocurrencia posible de n sucesos de la fuente original.

$$H(S^n) = n \times H(S)$$

Un código se dice *no singular* si todas las palabras de código son distintas.

Un código se dice *unívocamente decodificable* si para cualquier n, la extensión de orden n del código es no singular.

Un código se dice *instantáneo* si puedo decodificar cualquier secuencia sin conocer los símbolos que la siguen. Un código es instantáneo sii es libre de prefijos.

Sea L la longitud media de un código. Para que el código sirva para expresar unívocamente los eventos de una fuente S, entonces tiene que pasar:

$$H(S) \leq L$$

Un código se dice *compacto* cuando tiene longitud media menor que todos los códigos unívocos que sirven para representar la misma fuente.

3.3. Canales y ancho de banda

El ancho de banda de un canal se define como el rango de frencuencas alrededor del punto de amplitud máxima, donde para cada lado tomo hasta que la frecuencia deciende 3 dB, o aproximadamente a la mitad de la amplitud.

La capacidad de un canal de ancho de banda B y signal-to-noise-ratio SNR expresado como fracción (no en decibeles) es (Teorema de Shannon):

$$C = B \times log_2(1 + SNR)$$

Definimos el *decibel* como:

$$\left[\frac{V_{out}}{V_{in}}\right]_{dB} = 10 \times log_{10}\left(\frac{V_{out}}{V_{in}}\right)$$

Para poder reconstruir la señal de un canal de bandwidth limitado B tengo que samplear con frecuencia $f \geq 2 \times B$.

4. Capa Física

- Punto a punto: UTP.
- Acceso multiple: Coaxil / Satélite (poco alcance y/o son lentos)
- NRZ: Directo
- NRZI: Si viene un 1 cambio
- 4B/5B: Cada 4 bits los codifica en 5 bits mediante una tabla (solo se preocupa de que no haya muchos 0s consecutivos, total despues puede hacer NRZI que se ocupa de los 1s consecutivos sin pérdida.

- Manchester: 01 es 0 y 10 es 1, dos bits por bit.

Bit Stuffing: Delimitador 01111110. Si hay 5 1s consecutivos pongo un 0 atrás.

Dist. Hamming: (de 1 codificación) Max cant de bits de diferencia entre 2 códigos válidos.
Si dist hamming=D, se detectan D-1 bits erróneos y se corrigen $\lfloor \frac{D-1}{2} \rfloor$.

5. Capa Enlace

5.1. Ethernet (IEEE 802.3)

Dominio de colisión: Los que estan en mismo segmento ethernet (componente conexa delimitada por switches, no por hubs).

Paquete Ethernet:

- Preamble (64 bits)
- Dest (48 bits)
- Src (48 bits)
- Type (16 bits) > 1500 es un type, sino es length
- Body (Variable)
- CRC (32 bits)

5.2. Sliding Window

Ventana de k en el emisor significa que puedo tener hasta k mensajes no ACKeados al mismo tiempo (stop and wait es con $k = 1$). Ventana de k en el receptor es que puedo recibir y ACKear hasta k mas allá del primer no ACKeado que tenga (es básicamente el lugar en el *buffer*).

SWS = Ventana Sender
RWS = Ventana Receiver
S = Cant. números de secuencia
$SWS \leq S/2$. (Ver en resolución de exámenes la justificación).
Es óptimo que $SWS = RWS = S/2$. Si $SWS > RWS$, se pierden paquetes. Si $SWS < RWS$ nunca se usa el espacio que le sobra al receiver.
Preámbulo ethernet: Para sincronizar placas (64 bits de 01010101...)

Control de flujo: Con ACKs. Se puede decirle al receptor "banca que estoy con buffer lleno". Mensajes RR (Receiver ready) y RNR (Receiver not ready).

5.3. Wireless (IEEE 802.11)

Spread spectrum: Transmito en un rango de frecuencias (por redundancia)

Frequence hoping: Voy cambiando "al azar" de frecuencia (para evitar que me escuchen indeseablemente). Por supuesto, emisor y receptor deben tener mismo algoritmo de generación de random y misma semilla.

Direct Seq: Mandar xoreando con bits "al azar" (misma condición que antes para el "azar").

Terminal oculta: 3 nodos alineados. Los dos de la punta le hablan al del medio sin saber que está el otro y colisionan.
Terminal expuesta: A-B-C-D. B habla con A y como C lo escucha no manda a D cuando podría hacerlo sin drama.

Collision avoidance (MACA):
RTS: Request to send (incluye el largo de lo que quiero mandar). CTS: Clear to send. Data. Finish.
Es con MACA, collision avoidance. Primero el emisor envía un RTS indicando que quiere transmitir y detallando cuantos bytes. Si le llega un CTS (que también contiene la longitud del mensaje) transmite.

A su vez todos los que escuchan un CTS saben que estan en rango de un receptor y no pueden transmitir hasta el final de los datos (saben cuando es porque escucharon la longitud contenida en el CTS) porque ocasionarían colisión (esto evita el problema de terminal oculta). Por otro lado, si se ve un RTS se espera, y si no se ve el CTS, se puede transmitir (para que no se de el problema de terminal expuesta). MACAW agrega un ACK para confiabilidad y un algoritmo de Backoff.

Distributed system: Muchos Access Points conectados con cable (*backbone*) y cada moving host se conecta a uno de ellos (puede cambiar si quiere).
Probe, Probe Resp, Association request, Association Resp
Frame: Hasta 4 direcciones (src, dst, src AP, dst AP)

5.4. Datagramas vs Circuitos virtuales

Contención: Buffering porque el bandwith no me da para mandar todo lo que entra.
Congestión: Switch que no le alcanza el buffer (dropea cosas)

2 approaches: Datagrama (sin conexión, conmutación de paquetes) o Circuito Virtual (con conexión, conmutación de circuitos).
En cada switch una tabla de ruteo: Dest $\rightarrow$ port de salida. Es importante para la toleranacia a fallos que estas tablas puedan establecerse dinámicamente.

Virtual circuit:

- Setear circuito: Signalling ó manualmente. VCI: id del circuito (scope local a cada link)
- Los buffers de sliding window son por circuito (si no hay más buffers se deniega la creación de uno nuevo)
- Control de flujo hop by hop.

Source routing: Poner info de ruteo en el header (no es práctico, porque el source tendría que poder saber toda la ruta que va a hacer el paquete)

Brigde: Puente entre 2 ethernets para formar extended LANs.
Para crear tabla de ruteo puede matchear la MAC de source de cada paquete con el port por el cual la recibió (asociación que timeoutea por si el host cambia de conexión/subred). Si llega un paquete que no está en la tabla, se lo manda a todos.

Problema: Si hay loops en la LAN los paquetes dan vueltas para siempre.
Spanning tree: (solo necesita un arbol que llegue a toda la red, si algún switch no se usa no importa)

1. Elegir root (nodo de $<$ id, todos mandan el $<$ que vieron hasta ahora durante k segundos, como el tamaño de la red es limitado, esto asegura que al final todos saben el root).
2. Camino mínimo (DFS con reentrada, cada vez que cambio mi camino al root, vuelvo a enviar a mis vecinos con mi nuevo peso).
3. Para cada LAN se elige como designated bridge (punto de entrada al ST) el switch con camino mas corto o con $<$ id para romper empates.

Mensaje: (id_root,cam_min_root,id_sender)

BPDU: bridge protocol data unit.
VLAN: Identificada en header 802.1Q (STP). Los BPDU no lo tienen porque lo sacan de ese header.

6. Capa Red

Piggybacking: Mandar ACKs dentro de otros mensajes.

6.1. IP

Debe correr en todos los nodos.
Tipo de servicio: Datagrama / Mensajes / Paquetes. Best effort (trata pero no garantiza la llegada)

No garantiza llegada, ni órden, ni unicidad (cuando hay retransmisión, algunos paquetes pueden llegar 2 veces porque la primer emisión llegó tarde y dio TO de todas maneras).

Header (32 bits por item):

- Version, Header Len, TypeOfService, Length (bytes)
- ID fragmento, flags (MF, DF), offset (en 8-byte chunks, los fragmentos son multiplos de 8 bytes)
- TTL, Protocol (TCP,UPD,etc), Checksum
- SRC
- DST
- options (variable), padding (variable)
- Data (variable)

MTU: Maximum transmission data unit (máximo paquete que puede enviar un determinado protocolo L2)

Clases de direcciones (preamble/network/host):

a. "0"+7 bits+24 bits
b. "10"+14 bits+16 bits
c. "110"+21 bits+8 bits

Dir de red: Mask+Todo 0s. Dir broadcast: Mask+Todo 1s.

ARP: Busca la MAC de una IP dada. Se aprovechan los paquetes para asociar la IP a la MAC. Query: IP? El dueño de esa contesta con su MAC.

DHCP: servers 1 ó más (IPs disjuntas)

- Host hace un broadcast con DHCP server (esto solo queda dentro de su red)
- Relay agent hace un unicast al DHCP server (si esta fuera de la red, o sea, la red no tiene uno propio)
- Vuelve por el mismo camino (inverso) la respuesta.

6.1.1. Routing

Un router puede estar en varias redes. Forwardea paquetes según:

- Si alguna salida mia es la red de la dirección
- Si está en la tabla de ruteo
- Default gateway

RIP / distance vector: Bellman-ford. Problemas: Conteo a infinito. Split Horizon: No mandar al que me mandó (sólo la info que aprendi de él).
Split Horizon With Poison Reverse: Le mando infinito al que me mandó (sólo la info que aprendí de él). Si no, decir que $> k$ es equivalente a infinito.

OSPF / link-state: Hace Dijkstra, permite dividir por áreas, por eso es mas escalable.

6.1.2. ICMP

Errores de IP que se devuelven (no se asegura, es best effort).

- Unreachable host
- Timeout (TTL=0)
- Timeout en juntar fragmentos
- Echo (ping)
- Otros

6.2. MPLS

MultiProtocol Label Switching.

Para usar Internet en cosas que no soportan IP.

Cambia IP por labels, lo que facilita el lookup (solo es exact match en la tabla de ruteo).

Requiere configurar las tablas de ruteo para dichos labels. Sino, armar las rutas previamente automáticamente usando *downsteam* (empezar la ruta desde el destino) o *upstream* (empezar la ruta desde el origen).

7. Capa Transporte

7.1. UDP

Solo agrega a IP demultiplexación de puertos en los hosts y un checksum. No confiable sin conexión, es *best effort* como IP.

ID de un proceso: No se puede usar PID porque puede haber distintos sistemas operativos en la red. Se usa port o mailbox. El port tiene scope de máquina, la dirección completa es IP + port.

Formato de paquete UDP:

- SRC port (16 bits)
- DST port (16 bits)
- Length (16 bits)
- Checksum (16 bits)
- Data (variable)

Para comunicarse debe saberse a priori el puerto o establecer una especie de DNS para puertos, de modo de tener un sólo puerto fijo y el resto se consultan a través de éste.

La identificación de una conexión es <src-ip,src-port,dst-ip,dst-port> asi que un server puede recibir muchos clientes en el mismo puerto (un HTTP por ejemplo recibe a todos en el puerto 80) pero máximo una conexión por cliente en dicho puerto.

Checksum: Incluye header y pseudoheader (protocol number, IP addresses y length [1]).

7.2. TCP

Transport Control Protocol: Confiable con conexión. Se ve hacia afuera como un byte stream, no orientado a mensajes, aunque para adentro bufferea y manda las cosas como mensajes (ya que IP maneja mensajes). Mantiene el órden y es full-duplex. Adopta control de flujo y de congestión.

Control de congestión $\neq$ control de flujo (el primero es en toda la red y el segundo point-to-point).

[1] Si, length se suma 2 veces en el checksum

Sliding window:
Problemas adicionales a los de la capa de enlace y sus necesidades de solución:

- No hay link dedicado: Fase de conexión.
- Puede haber reordenamiento de paquetes: Maximum Segment Lifetime.
- Camino y por lo tanto RTT variable: TimeOut adaptativo.
- Congestión: Control de congestión.
- Ventana máxima variable: Ventana adaptativa (control de flujo).

hop-by-hop guarrantee NO implica end-to-end guarrantee (los nodos pueden flashear, aunque los ejes todo bien)

Formato de paquete TCP (32 bits por item):

- SRC-port, DST-port
- SeqNum
- ACK
- headlen, 0, flags, AdWindow (Flags: SYN, FIN, ACK, RST, URG, PUSH)
- Checksum (incluye pseudo-header, como en UDP), UrgPtr
- Options
- Data (variable)

Establecimiento de la conexión: 3-way handshake.
SYN, SYN+ACK, ACK. Activo (client) y pasivo (server) generan su número de secuencia inicial al azar (para evitar colisiones con una conexión previa (otra encarnación) cerrada hace muy poco) y ACKea con el número recibido + 1 (el número de ACK indica el próximo byte esperado, no el último recibido).

Control de flujo: AdvertisedWindow, se usa el campo para decir cuanta ventana para sliding window tiene el receptor disponible. Si el receptor lee lento, $AdWin$ se va a cero y detiene al emisor.

$$AdWin = MaxRcvBuffer - ((NextByteExpected - 1) - LastByteRead)$$

y del lado del emisor (hay que restarle los datos en vuelo):

$$EfWindow = AdWin - (LastByteSent - CantBytesACKed)$$

Si $AdWin$ se reduce a cero se debe bloquear al proceso que escribe para que no overfloodee el buffer de salida y continuar enviando 1 byte de modo de enterarnos cuando la ventana se abra de nuevo (el receptor es pasivo, solo responde a mensajes, no envia notificación de que reabrió su ventana). Se hace asi para mantener el lado cliente minimal.

La transmición se hace cuando (Transmission triggers):

- $>$ MSS bytes en buffer (Max Segment Size, gralmente = MTU - Headers)
- Explícitamente pedido por la app (PUSH)
- Timer (ver más abajo, no es tiempo de reloj común)

Silly window syndrome: Muchos paquetes chiquitos (la ventana queda siempre poco abierta)

- RCV puede esperar a tener un tamaño razonable (cerca de un MSS) para reabrir la ventana
- RCV puede delayear ACKs para "juntar" paquetes chiquitos y evitar introducir "contenedores pequeños" (ver analogía en el Peterson) en el sistema

Timers o algoritmos para transmisión:

- Algoritmo de Nagle: Mando si puedo 1 MSS, si no hay nada en tránsito o luego de un ACK de cualquier cantidad
- Clock: Timeout es una función del RTT
  - Estimar RTT con el promedio hasta ahora ponderado para los más cercanos (descuento exponencial). Luego `RTO=2*RTT`.
  - Karn/Partridge: En las retransmiciones no estimar (ya que no se sabe a cual transmisión corresponde el ACK), simplemente `RTO*=2`.
  - Jacobson/Karels: El anterior y en lugar de `RTO=2*RTT` hacer `RTO=RTT+4*desvio` donde el desvio también se va estimando.

Slow start: Empiezo "de a poco" con el CWND=1, aumentando exponecialmente en cada ACK.

Fast Retransmit: Si llegan 3 ACKs duplicados (4 iguales en total) retransmito aunque no haya habido time out.

Fast Recovery: Cuando se activa fast retransmit divido por 2 el CWND en lugar de mandarlo a 1. Se infla la CWND para tomar en cuenta los ACKs recibidos (ya que son paquetes que no están mas en vuelo). Luego debe desinflarse al volver a la normalidad (para evitar contarlos 2 veces). REVISAR

Idle cycle: Si pasa un RTT sin hacer nada, vuelvo el CWND a 1 y vuelvo a Slow Start porque mi información acerca de la red esta desactualizada.

Resumen de etapas de control de congestión:
Empieza en SS → CWND > SSThres → Congestion Avoidance (Additive Increase / Multiplicative Decrease)
Si hay Time Out: Se divide SSThres a la mitad y CWND=1, volviendo a SS.
Si hay idle time: CWND=1 y volver a SS.
Si hay Fast Retransmit: CWND/=2 e inflar CWND con los ACKs de más (luego desinflarla). REVISAR

DecBit: Flag en el header TCP. Si llegan mas del 50 % con DecBit activado multiplico `CWND*=0.875`, sino `CWND++`;

RED (Random Early Detection): Dos thresholds α y β. Si el tamaño de la cola es $< \alpha$ dejo pasar, si es $> \beta$ no dejo y en el medio descarto con probabilidad proporcional a tamaño$-\alpha$.

Source Based Congestion:
`if (currWnd-oldWnd)*(currRTT-oldRTT)>0 then CWND-=0.125 else CWND+=MSS`.

TCP Tahoe: Slow Start y Fast Retransmit (sin Fast Recovery)
TCP Reno: Slow Start, Fast Recovery y Fast Retransmit
TCP Vegas: Calculo el bandwith con mi CWND y sampleo el real. Si hay mucha diferencia, ajusto para el lado correspondiente.

7.3. Taxonomía

- Lazo abierto/cerrado: Es cerrado si hay retroalimentación del modo de control sensando el medio.
- Origen/destino: Quién hace el trabajo de control.
- Reservation/feedback: Prealocar espacio vs sensar el medio y en base a eso acomodar el comportamiento
- Feedback explícito/implícito: Mensajes de control con el feedback o deducirlo según el comportamiento

7.4. SCTP (RFC 2960)

Multistreaming: Permite mandar varios chunks (mensajes) en un mismo paquete de nivel 3, optimizando el tráfico. Usa ACK selectivo.

4-way handshake: Previene ataques de Denial Of Service (A los que TCP es muy vulnerable) ya que para poder hacer que el server aloque recursos tiene que el cliente alocarlos tambien (ya que solo se aloca luego del 2do mensaje, que contiene una cookie que permite verificar correctitud).

Multihoming: Muchos IPs en un mismo host, se avisa mediante paquetes de control. Retransmite cambiando de IP de la transmisión original (para que vaya por otro camino). Puede determinar que alguna de las IPs se volvió *unreachable* (últimos 6 *heartbeats* no contestados) y continuar utilizando la otra como principal.

Verification Tags: Asegura que no se mezclan paquetes de distintas encarnaciones de la misma conexión.

Otros: Explicit congestion notification. Cierre no asimétrico (distinto de TCP).

Esta bueno para FTP porque haces todo en la misma conexión.

8. Capa Aplicación

8.1. DNS

Mapea nombres de dominio a valores. Coleccion de registros: <name, value, type, class, TTL>.
Types: A:Address, NS: Name Sever, MX: Mail Exchange (server de mail para esa zona), CNAME: Cannonical name (para hacer aliases).
Class: Solo existe IN (Internet) hasta ahora.
TTL: Tiempo de expiración del registro.

DNSs secundarios:

- Zone transfer: Transferencia de información desde el primario que corresponde. Solo se transifere si hay diferencias (dependiendo de la versión se transifere todo o solo las diferencias). Se hace cada tanto (un par de horas en general).
- Serial number: Numero de serie. Cuando el primario se modifica se incrementa (se usa para que el secundario sepa si hubo cambios o no)

Autoritativo: Es un DNS de esa zona (primario o secundario)
No autoritativo: Es un dato cacheado.

Mecanismos de resolución:

- Iterativo: El cliente pregunta y le devuelven un registro o un puntero a un nuevo server, con lo cual debe repreguntar.
- Recursivo: Transparente para el cliente, cada server se hace responsable de toda la query que se le hace.
- Hay una versión híbrida en la cual cada host cliente habla solo con un proxy y el proxy hace iterativo. Si los hosts no tienen salida a internet, esto minimiza mucho el tráfico interno respecto de un iterativo directo (y aparte maximiza el uso de cache del DNS en el proxy, porque todos los hosts la comparten).

8.2. HTTP

Comandos importantes: `GET, HEAD`. (El primero trae un recurso y el segundo solo los headers).

HTTPS: HTTP sobre TLS.

Versión 1.0: Conexión TCP no persistente, la cierra luego de un comando.
Versión 1.1: Conexión TCP persistente, permite bajar varios recursos en la misma.

8.3. FTP

Activo: Cuando quiero bajar algo tengo que abrirle un puerto y el se conecta ahí para mandarme las cosas. Hay una conexión que la abre el cliente (la de control) y el resto las inicia el server.
Pasivo: El server abre puertos para cada resultado de un comando. El cliente inicia todas las conexiones (sirve para que los firewalls no molesten).

8.4. SMTP y ESMTP

Comandos: `HELO, MAIL, RCPT, TURN`. TURN esta bueno para cuando tenés un server que no es muy fijo y entonces nadie se le puede conectar. Te conectas vos y luego hacés TURN para recibir (es casi como avisar que estas ahí y que el otro se conecte).

ESMTP resuelve problemas de seguridad (agrega autenticación al protocolo, `ETRN`).

8.5. RTP

Real-time Transport Protocol. Sirve para streaming de multimedia, en gral va sobre UDP. Agrega un timestamp, ID de codificación que identifica el tipo de dato y control de congestión.

9. Seguridad

Claves simétricas (misma clave y algoritmo encripta y desencripta) y asimétricas (clave pública y privada).

Asimétricas:

- RSA: Dados 2 primos grandes se calcula $N = (p-1) \times (q-1)$. Sean dos numeros d, e tales que $d = e^{-1}\ mod\ N$. Clave privada $= < d, N >$ y pública $= < e, N >$.
- DSA, ElGamal (Sirven para *digital signatures*).

Simétricas

- DES, triple DES, César, one-time-pad, blowfish

SSL: Secure socket layer.
TLS: Transport layer security, sucesor de SSL.

SSH: Secure shell (no es un protocolo de transporte sino una aplicación).

10. Exámenes resueltos

10.1. 1ros Parciales

10.1.1. 1er cuat 2006, 1er parcial

Prácticos

1. Hay que hacer 2 subredes 200.11.160.0/25 y 200.11.160.128/25. Eso deja 126 direcciones libres para cada red (quitando broadcast y la dirección de la red), 70 serán para los hosts y 2 para el router, una para la interfase hacia adentro y otra hacia afuera.
2. *a*) Manda por la interfase 1 a un host.
   b) Droppea el paquete porque es fruta el 256
   c) Se reporta que no se puede alcanzar el destino
   d) Se reporta que no se puede alcanzar el destino
   e) Manda por la interfase 1 a un host.
3. Porque el problema sería la falla en detectar colisiones, pero esto solo se produce con paquetes pequeños (menores a RTT$\times$*bandwidth*) que no estan sucediendo.
4. Que asignen conjuntos de IPs disjuntos, ya que DHCP no provee mecanismos de sincronización entre distintos servers.
5. 200 Mbps

Conceptuales

1. No, porque la función debe ser biyectiva y esto es imposible por la cardinalidad de ambos conjuntos. La longitud media de un código es mayor o igual a la entropía de la fuente, que en este caso puede llegar a ser de $\log 256 = 8$, con lo cual no es posible para cualquier entrada cumplir el objetivo de reducir los 8 binits/símbolo.
2. El receptor puede enviar notificación al emisor del lugar que tiene (Mensajes RR y RNR).
3. A que cubren distintos tipos de necesidades según las características del medio.
4. Porque permite dividir por areas el ruteo y hacerlo jerárquicamente.

10.1.2. 1er cuat 2006, 1er recu

Prácticos

1. Bit de more fragment (MF) en 1 o luego offset $\neq 0$.
2. Puede ser global o por VLAN. Hacerlo por VLAN ocupa mas ancho de banda, pero optimiza el uso de todos los links, lo que es beneficioso a la larga por la reducción de las colisiones y la posible reducción de los caminos. Tambin se puede hacer por grupos de VLANs, tratando de capturar lo mejor de ambos mundos.
3. $$delay = 36000km/c = 0{,}12s$$
$$bandwidth = 1Mbps$$
$$delay \times bandwith = 0{,}12s \times 1Mbps = 0{,}12Mb$$
$$frames = 0{,}12Mb/1250B = 0{,}12Mb/10000b = 12{,}58$$
4. Cuadrito NO HECHO
5. Envia un ICMP avisando del error como cualquier otro mensaje con TTL excedido.

Conceptuales

1. La entropía define la cantidad media de información por símbolo. Es máxima cuando todos los símbolos son equiprobables. Para que la entropía máxima sea de 1 bit por símbolo hacen falta exactamente dos símbolos (y dicho máximo se alcanzará cuando ambos tengan probabilidad 1/2).
2. Split horizon y asumir que una distancia $> k$ es equivalente a infinito.
3. Stop and wait garantiza que no vas a llenar el buffer ya que solo hay 1 paquete en vuelo cada vez y solo se envia un segundo paquete cuando el primero fue procesado y ackeado.
4. Sin conexión y no confiable.

10.1.3. 1er cuat 2005, 1er recu

1. *a*) Si, ya que el problema de terminal oculta se puede seguir dando. Si los AP están suficentemente lejos, el problema de terminal expuesta puede evitarse.
   b) Son 4, el origen, el AP de origen, el destino y el AP de destino.
2. Buenos Aires: 192.168.1.0/25
   VLAN 1 (30 hosts): 192.168.1.0/27
   VLAN 2 (30 hosts): 192.168.1.32/27
   VLAN 3 (10 hosts): 192.168.1.64/28
   Tucumán: 192.168.1.128/25
   VLAN 1 (20 hosts): 192.168.1.128/27
   VLAN 2 (20 hosts): 192.168.1.160/27
   VLAN 3 (20 hosts): 192.168.1.192/27
   Esto deja suficiente lugar para las interfases de los routers (ya que a partir de la .224 en Tucumán y de la .80 en Buenos Aires estan todas libres).

3. *a*) No, es la cota mínima. No hay cota máxima, dada una codificación puedo agregar el mismo símbolo al final de cada código y hacer una con longitud media mayor.
   b) Si, dado un código instantaneo puedo hacer lo que dije antes y queda un instatáneo no óptimo (ya que es peor que el original).
4. Puede ser todo, dado que hay 4 flujos y todos tienen 1 Gb de capacidad. [2]
5. Dado que solo tiene 2 conexiones, lo que viene por una lo manda siempre por la otra y listo.
6. Nada. El TTL se reduce normalmente en cada hop.

10.1.4. 2do cuat 2004, 1er parcial

1. - Per-VLAN (un spanning tree por cada VLAN)
   - Multiple spanning tree (hace un spanning tree por cada grupo de VLANs)
   - Spanning tree normal (hace un único spanning tree para toda la red)
2. - Primer piso: 193.168.0.0/27
   - Segundo piso: 193.168.0.32/28
   - Tercer piso: 193.168.0.48/26
   - Cuarto piso: 193.168.0.112/26
   - Quinto piso: 193.168.0.176/28
   - Servers: 193.168.0.192/28
   - Router1: 193.168.0.208
   - Router2: 193.168.0.209
3. - No tiene conteo a infinito.
   - Es más escalable porque agrupa por áreas y la jerarquización ayuda a la escalabilidad.
   - Se recupera más rápido de las caidas.
4. No, es falsa. Si tengo dos símbolos equiprobables, hay solo dos códigos óptimos.
5. $$emisor + receptor \leq CantSeqNum = 8$$
   $$emisor = receptor$$
   $$emisor + emisor \leq 8$$
   $$emisor = 4 = receptor$$
6. 1 Se droppea.
   2 Se avisa por ICMP (mensaje Destination Unreachable).

10.1.5. 1er cuat 2001, 1er recu

1. *a*) Dos bits para cada símbolo son suficientes: $\Pi = 00$ $\otimes = 01$ $\nabla = 10$ "espacio en blanco" $= 11$
   b) Haciendo Huffman da óptimo: $\Pi = 10$ $\otimes = 110$ $\nabla = 111$ "espacio en blanco" $= 0$
2. En ambos casos va a mandar 17 tramas (en 16 entraría justo si fueran todos datos, y como tiene poco overhead con una trama extra alcanza para lo que faltó). Si contamos el overhead sumando el padding de la última trama en ambos nos queda $(17 \times 64KB) - 1MB/(17 \times 64KB) \approx 0,06$. ?????
3. *a*) No vimos NO HECHO
   b) Si, Ethernet no tiene manejo de prioridades ?????
   c) No, los hubs forwardean todo asi que la saturación se mantendrá. La papa es poner switches.
4. No vimos NO HECHO
5. No vimos NO HECHO

[2] Buscar diferencia entre Ehterchannel y 802.1Q

10.1.6. 2do cuat 2004, 1er parcial

1. Por VLAN, en general o combinadamente por grupos de VLANs.

2. *a*) La codificación NRZ codifica cada bit en un bit, por lo cual la longitud total a enviar es $1024\ B = 1\ KB$ y tarda $1\ KB/256\ Kbps = 1\ KB/32\ KBps = 1/32\ s = 0,03125\ s$.

   b) Ídem $2a$

   c) La codificación 4B/5B codifica cada 4 bits en 5 bits, por lo cual la longitud total a enviar es $1024\ B \times 5/4 = 1280\ B$ y tarda $1{,}28\ KB/256Kbps = 1{,}28\ KB/32\ KBps = 0,04\ s$.

   d) En Manchester se codifica cada bit como 2 bits, por lo cual va a tardar el doble que en $2a$, o sea, $1/32\ s \times 2 = 1/16\ s = 0,0625\ s$.

3. - Laboratorios con internet: 157.92.26.32k/27 (para cada k entre 0 y 4)
   - Laboratorios sin internet: 192.168.0.32k/27 (para cada k entre 0 y 4)
   - Secretaría y cuartitos: 157.92.26.160/26
   - Dirección de la interfase hacia adentro del router: 157.92.26.225
   - Dirección de la interfase hacia afuera del router: 157.92.26.226

4. Verdadero, existe al menos 1 dado por el algoritmo de Huffman. Ahora, dado ese código se puede agregar a cada código el mismo símbolo repetido k veces para cualquier k al principio y claramente lo que queda sigue siendo libre de prefijos.

5. Split horizon, split horizon with poison reverse y upper bound en la distancia.

6. Se puede hacer un envio directo a la MAC del server. Si suponemos que se tiene la MAC del server pero se desea enviar un paquete IP, primero se hace un RARP para conseguir la IP, luego DHCP para conseguir una IP para uno mismo y luego se lo contacta por IP.

10.1.7. 1er cuat 2004, 1er parcial

1. Con STP por VLAN o STP global.

2. *a*) 011110111110011111010

   b) Supongamos que el emisor tiene una ventana de e y hay S números de secuencia. Mas aún, supongamos que $e > S/2$, ya que es en esos casos en los que se da el solapamiento. Se mandan e paquetes. Sin importar el tamaño de la ventana del receptor, supongamos que procesa todos los mensajes rápido de manera que siempre tiene lugar en el buffer y que envía todos los ACKs, pero éstos se pierden. Luego del time out en el emisor, este reenviará e paquetes, pero uno de éstos paquetes tiene el mismo número de secuencia que uno de los anteriores (porque $2e > S$) y el receptor piensa que es el nuevo mensaje en lugar de dropearlo por ser una retransmisión, que es lo que debería pasar.

3. *a*) Lo envía por la interfase 1.

   b) Lo envía por la interfase 0.

   c) Lo envía al router 2.

   d) Es un paquete inválido por el 256.

   e) Lo envía al router 1.

   f) Lo envía al router 2.

4. No se puede detectar la colisión en wireless, por ejemplo en el problema de términal oculta ninguno de los 2 emisores sabe que esta colisionando, y el receptor no recibe ninguno de los 2 mensajes.

5. El único problema es que se precisaría un buffer que puede ser muy grande para decodificar, pero no deja de ser posible hacerlo.

6. El DHCP server debería asignar la IP según la subred de la cual le vino el mensaje (si es un broadcast es su propia red y si es un unicast de un relay agent es la red de éste). El RFC no requiere que esto sea así, y deja la decisión al servidor.

10.2. 2dos Parciales

10.2.1. 1er cuat 2003, 2do parcial

1. No vimos NO HECHO
2. Hay que encapsular UDP sobre TCP. La forma mas cómoda es que el server de tunneling desencapsule el paquete UDP del envio TCP que le llega y se lo envie a si mismo como UDP.
3. Si se pierde algo podría no activarse Fast Retransmit y por lo tanto tampoco Fast Recovery. Otro problema es que puede perturbar la medición y estimación del RTT.
4. Hacer que el crecimiento de CWND sea proporcional a la longitud que se ACKeó.
5. 7 conexiones en HTTP 1.0 y 3 en HTTP 1.1.
6. Muchos servers en edificios distintos. Uno es primario y sobre él se hacen las actualizaciones y el resto son secundarios y cada cierto tiempo actualizan haciendo un zone transfer desde el primario. Esto último puede querer decir copiar todo o solo las diferencias (siempre en caso de que haya alguna diferencia, si no no se hace nada) según la versión de DNS. Chamuyar un poco sobre balanceo de carga.

10.2.2. 1er cuat 2005, 2do parcial

1. Se conecta al servidor POP3, en el medio se resuelve la dirección pop3.inta.gov. Después pone comando `user prueba` que quiere decir que quiere loguearse como prueba, y después con el comando `pass xxxxx` pone su password que es "xxxxx". Luego lista todos los mails que tiene en su cuenta. Tiene dos: uno que pesa 14KB y otro que pesa 3KB. Por último termina la sesión con el comando `quit`.
2. *a*) Para hacer la estimación se podría utilizar un promedio, pero en este caso sirve más usar la media ponderada porque le da más importancia a las últimas mediciones. Esto es importante porque el RTT va variando con el paso del tiempo.
   b) Ninguno, como dice Karn, en lugar de estimar, en ese caso simplemente se duplica el RTO.
3. Jacobson y Righetti dicen que es lazo cerrado, porque se usa el tamaño de la cola del router para aproximar la congestión en la red. Sin embargo se puede pensar este sistema como un sistema de lazo abierto, pues los umbrales en base a los cuales se toma la decisión están fijos.
4. Porque puede haber ciclos en la red, y el proposito es que los paquetes no se queden dando vueltas para siempre.
5. *a*) Asimétrica.
   b) Los protocolos que usan certificados digitales son TLS, SSH, Diffie-Hellman.
   c) En la dificultad de factorizar primos, lo que garantiza todos los pasos intermedios de la cadena, y en la confianza en un tercero (VeriSign) que garantiza el primer paso.
6. FTP podría funcionar en modo pasivo, ya que en ese caso todas las conexiones se establecen en el sentido cliente $\rightarrow$ servidor y no son bloqueadas.

10.2.3. 1er cuat 2005, 2do recu

1. Si, si los hosts están cerca y el RTO estimado es menor a 500ms, es bastante factible que no lleguen los ACKs duplicados antes del timeout y nunca se usa Fast-Retransmit.
2. Closed-loop, feedback explícito y source based (porque quien toma la acción para evitar el problema es el origen). Sin embargo, el destino también tiene un poco de injerencia ya que con los ACK se controla al emisor, y el destino puede decidir usar delayed ACKs o no. (Esto me lo respondió Vukovic).
3. Se usa downstream por default para armar paths a los hosts que están fuera de la red MPLS iniciando en el edge router al que están conectados.

4. Saber a priori el tamaño para mostrar una barra de progreso. Si cacheas la página, revisar primero la fecha y solo hacer un GET si la fecha de modificación es distinta a la cacheada.

5. Si a ambas. Para la misma zona no es útil, para distintas zonas es normal.

6. Con 3 paquetes: → FIN, ← ACK+FIN, → ACK.
   Con 4 paquetes: → FIN, ← ACK, ← FIN, → ACK.

10.3. Finales

10.3.1. 16/12/2004

1. No tiene conteo a infinito. Es escalable por el uso de áreas. Tiene autenticación de mensajes. Tiene balanceo de carga. Asigna peso a los ejes, por lo cual descubre el camino mínimo en tiempo, usar menos hops no siempre es óptimo (puede requerir usar hops muy lentos en vez de un poco más, pero rápidos).

2. Dropea el paquete. Avisa por ICMP con unreachable host.

3. Puede utilizar SYN cookies: encodear en el número de secuencia la información que necesita persistir y sólo alocar recursos en el ACK final. Esto obliga al emisor a tener que responder al segundo mensaje, lo cual lo hace alocar recursos y es mas costoso hacer un ataque (ya no es un blind attack).

4. Si hay un time out, si hay 3 ACKs duplicados en una implementación sin Fast Recovery (tahoe) o luego de un ciclo ocioso.

10.3.2. 11/08/2005

1. No tiene conteo a infinito. Es escalable por el uso de áreas. Tiene autenticación de mensajes. Tiene balanceo de carga. Asigna peso a los ejes, por lo cual descubre el camino mínimo en tiempo, usar menos hops no siempre es óptimo (puede requerir usar hops muy lentos en vez de un poco más, pero rápidos).

2. Dropea el paquete. Avisa por ICMP con unreachable host.

3. Puede utilizar SYN cookies: encodear en el número de secuencia la información que necesita persistir y sólo alocar recursos en el ACK final. Esto obliga al emisor a tener que responder al segundo mensaje, lo cual lo hace alocar recursos y es mas costoso hacer un ataque (ya no es un blind attack).

4. Si hay un time out, si hay 3 ACKs duplicados en una implementación sin Fast Recovery (tahoe) o luego de un ciclo ocioso.

5. La entropía es la cota mínima para la longitud media de código.

10.3.3. 20/07/2006

1. Tenés más problemas: Reordenamiento de paquetes, RTT variable, no hay link punto a punto, congestión y tamaño de ventana variable.

2. Jerarquización, flexibilidad, tolerancia a fallos. REVISAR

3. Que el coeficiente de fairness da cercano a 1, es decir, en castellano, que muchos flujos que comparten un link lo hacen de forma pareja, evitando la inanición. Si hay n flujos en un link dado, cada uno usará cerca de $1/n$ del ancho de banda disponible.

4. Es con MACA, collision avoidance. Primero el emisor envía un RTS indicando que quiere transmitir y detallando cuantos bytes. Si le llega un CTS (que también contiene la longitud del mensaje) transmite. A su vez todos los que escuchan un CTS saben que estan en rango de un receptor y no pueden transmitir hasta el final de los datos (saben cuando es porque escucharon la longitud contenida en el CTS) porque ocasionarían colisión (esto evita el problema de terminal oculta). Por otro lado, si se ve un RTS se espera, y si no se ve el CTS, se puede transmitir (para que no se

de el problema de terminal expuesta). MACAW agrega un ACK para confiabilidad y un algoritmo de Backoff.

5. Es entre el 22 y el 0. Se puede notar que la onda desde el 0 corresponde a un RED ya que tiene subidas y bajadas abruptas entre 3 niveles mas bien estables. Cada nivel corresponde a uno de los estados del RED (pasar siempre, no pasar nunca o pasar con probabilidad p).

10.3.4. Oral Persona X 10/2006

1. ¿Qué es un modem?
   Un aparato que permite mandar una señal digital sobre un medio análogico.

2. ¿Qué es entropia? Es la esperanza de la información. Es cota mínima para la longitud media de código.

3. ¿Dos personas distintas agarran un documento cualquiera y obtienen la misma entropia? Si la codificación la tomamos fija, si.

4. Congestión en TCP
   Slow start, congestion avoidance, fast retransmit, fast recovery.

5. ¿Cuál es el problema de distance vector y 2 formas de evitarlo?
   Conteo a infinito. Se evita con split horizon, split horizon with poison reverse o upper bound.

6. ¿Cuáles son los problemas de 802.11 y como los soluciona CSMA/CA?
   No podes detectar colisiones (terminal oculta). Haciendo MACA evita las colisiones pidiendo permiso (para detalles ver en el resúmen). La terminal expuesta tambien se soluciona con MACA.

7. Si tengo una empresa en dos edificios distintos que quedan a varios KM, ¿Qué puedo usar para hacer una LAN? ¿Qué protocolos intervienen? Si no tenés un dispositivo, creálo
   Internet. Tunneling. Ethernet sobre IP (o sobre TCP o sobre HTTP, según que te permita tu ISP)

8. ¿Cómo hacer una comunicación segura sobre IP?
   IPSec (agregar headers con firma digital). Sino, IP sobre TLS o IP sobre SSH con port forwarding.

9. ¿Cómo implementas un protocolo de nivel 3 con conexión sobre IP?
   Agrego un handshake y un protocolo de finalización de conexión. Otra es hacer IP sobre TCP.

10. ¿Cómo se firma un mensaje con firma digital?
    Le haces MD5 y encriptas el MD5 con tu clave privada (se puede no hacerle MD5, pero es muy lento hacerle RSA a todo).

11. ¿Para qué sirve ESMTP y que problemas soluciona?
    Soluciona problemas de seguridad, SMTP no tiene autenticación.